Perancangan Deteksi Kemiripan pada Abstrak Artikel Ilmiah Informatika

Reza Pahlevi[1)], Mira Ziveria[2)]

Informatika, Fakultas Industri Kreatif, Institut Teknologi dan Bisnis Kalbis
Jalan Pulomas Selatan Kav. 22, Jakarta 13210
[1)] *Email*: *rezaapahle@gmail.com*
[2)] *Email:mira.ziveria@kalbis.ac.id*

Abstract*: The purpose of this study is to develop a similarity detection application using cosine similiarity calculations which can be useful to find out the percentage of similarity of a text. Detection and calculation is done by cosine similiarity algorithm and framework flask with the help of the NLTK and Literature library which helps in proprocessing data to convert words that have affixes into standard words. Application development using the SDLC method. At the design stage the application detects and calculates the percentage of word-for-word similarities with cosine similiarity calculations. At the implementation stage cosine similiarity calculation will calculate the similarity of the text with the text. At the testing stage the cosine similiarity calculation will detect the similarity of words obtained from web forms built with the flask framework. The application can detect similarities in the text and display the similarity percentage.*
Keywords: *abstract, cosine similiarity, preprocessing, scientific articles, similarity detection*

Abstrak: *Tujuan dari penelitian ini adalah untuk membangun aplikasi deteksi kemiripan dengan menggunakan perhitungan cosine similiarity yang dapat bermanfaat untuk mengetahui besar persentase kemiripan sebuah teks. Pendeteksian dan perhitungan dilakukan dengan algoritma cosine similiarity dan framework flask dengan bantuan library nltk dan sastrawi yang membantu dalam proprocessing data untuk mengubah kata yang memiliki imbuhan menjadi kata baku. Pengembangan aplikasi menggunakan metode SDLC. Pada tahap perancangan aplikasi mendeteksi dan menghitung persentase kemiripan kata per kata dengan perhitungan cosine similiarity. Pada tahap implementasi perhitungan cosine similiarity akan menghitung kemiripan teks dengan teks. Pada tahap pengujiain perhitungan cosine similiarity akan mendeteksi kemiripan dari kata yang didapat dari form web yang dibangun dengan framework flask. Aplikasi dapat mendeteksi kemiripan yang terdapat pada teks dan menampilkan persentase kemiripan.*
Kata kunci: *abstrak, artikel ilmiah, cosine similiarity, deteksi kemiripan, preprocessing*

I. PENDAHULUAN

Plagiarisme atau sering disebut dengan kata plagiat berasal dari bahasa inggris *plagiary* dan bahasa latin *plagiarius* yang memiliki arti penjiplak atau pencuri. Sehingga plagiarisme memiliki makna yaitu perbuatan menjiplak ide atau karya seseorang yang akan diakui sebagai karya sendiri tanpa menyebutkan sumber ide atau gagasan tersebut berasal. Sehingga menyebabkan kesalahan pada ide atau gagasan tersebut bersumber dan siapa yang memiliki ide atau gagasan tersebut[1].

Pada proses pengerjaan tugas atau dalam menulis karya ilmiah dalam lingkungan perkuliahan dan dalam lingkungan penelitian membutuhkan referensi untuk membantu dalam penulisan tersebut namun terkadang terdapat penulis atau peneliti yang tidak menyebutkan sumber.

Plagiarisme sendiri merupakan tindakan mencuri gagasan atau hasil

pemikiran dan tulisan orang lain yang digunakan dalam tulisan seakan – akan gagasan atau tulisan orang lain tersebut ialah gagasan atau tulisan sendiri, sehinggua merugikan orang lain[2]. Plagiarisme ditemukan dalam aktivitas seseorang saat menjalankan masa pembelajaran atau dalam pekerjaan. Plagirisme merupakan kecurangan dari seseorang yang mencuri atau mengikuti materi orang lain tanpa mengutip narasumber tersebut.

Dalam kasus ini, *Natural Language Processing* digunakan untuk memproses data teks untuk mendapatkan akurasi, seberapa besar persentasi dari suatu penulisan merupakan kesamaan atau tidak. Dengan perancangan deteksi kesamaan asbtrak artikel ini, diharapkan tindak kesamaan dapat dikurangi dan menekankan kepada berbagai penulis untuk memberi narasumber jika susunan kalimat yang tertulis adalah sebuah kutipan.

Seiring dengan pesatnya arus informasi dan tidak terawasinya karya ilmiah yang dapat saja hasil dari kesamaan atau hanya *copy-paste* seperti kasus-kasus di Indonesia yang terjadi seperti pada kasus dugaan plagiarisme di program doktoral Universitas Negeri Jakarta (UNJ)[3]. Dalam kasus tersebut, deteksi kesamaan dapat meminimalisir kemungkinan terjadinya kesamaan isi dari suatu penulisan ilmiah atau karya tulis. Suatu karya tulis yang teridentifikasi sebagai plagiarisme tidak dapat *dipublish* dan diakui oleh berbagai institusi maupun masyarakat.

Deteksi kemiripan teks telah dilakukan pada beberapa penelitian sebelumnya, Dwi Susanto, achmad Basuki, Prada Duanda menerapkan metode Naïve Bayes pada kasus Deteksi Plagiat Dokumen Tugas Daring Laporan Praktikum Mata kuliah Desain Web dengan Metode Naïve Bayes[4] penelitian ini mengerjakan sebuah program pendeteksian plagiarism untuk digunakan pada pengumpulan laporan-laporan mata kuliah yang ditentukan untuk menghindari adanya tindak tersebut dan memudahkan pengajar untuk mengetahui laporan yang hasil plagiat atau tidak. Zudha Pratama, Ema Utami, M; Rudyanto Arief pada kasus Analisa Perbandingan jenis N-gram Dalam Penentuan *Similiarity Text* pada Deteksi Plagiat [5].

Penelitian yang telah dijelaskan secara singkat memberikan kesimpulan bahwa plagiarisme terjadi dalam penulisan jurnal-jurnal atau artikel-artikel ilmiah yang bertujuan mengedukasi namun apabila terjadi plagiarisme dalam proses penulisan hasil penelitian tersebut akan mengurangi dan menghilangkan kredibilitas peneliti dalam penelitian tersebut.

Maka, berdasarkan latar belakang tersebut peneliti ingin mengetahui bagaimana mengimplementasikan perhitungan *cosine similiarity* untuk mendeteksi kemiripan pada abstrak artikel ilmiah?

II. METODE PENELITIAN

Penelitian ini akan menggunakan perhitungan *cosine similiarity* sebagai perhitungan kemiripan sebuah teks dan menggunakan format txt untuk memasukan abstrak artikel ilmiah pada program untuk menghitung seberapa besar kemiripan yang terjadi pada abstrak tersebut dengan teks abstrak artikel ilmiah yang dijadikan sebuah acuan untuk menentukan abstrak sebelumnya seberapa besar dengan teks abstrak artikel ilmiah acuan dan ini sedikit berbeda dengan penelitian terdahulu yang menggunakan perhitungan *cosine similiairity* juga namun berbeda dalam *input* data teksnya dikarenakan penelitian tersebut menggunakan *file* berformat PDF dan aplikasi yang akan dibangun berbasis *website* dengan *user interface* yang sederhana dan tanpa biaya apapun untuk menggunakannya sehingga memudahkan pengguna dalam

menggunakannya secara gratis agar membantu penulis ataupun peneliti lainnya dalam penulisan agar tidak terjadi plagiarisme tidak sengaja yang dapat terjadi tanpa diketahui oleh penulis namun dapat merugikan penulis pada kemudian hari.

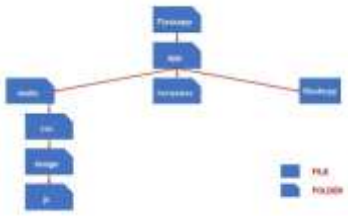
Pada *user interface* yang akan dirancang untuk pembangunan aplikasi berbasis *website* kesamaan ini sangat sederhana dengan hanya terdapat dua kotak teks untuk memasukan teks atau kalimat yang ingin di uji seberapa kemiripannya lalu terdapat satu kotak area teks untuk memasukan kata atau kalimat yang menjadi acuan dalam seberapa persentase kemiripan sebuah teks atau kalimat yang berada pada kota teks uji sebelumnya dan dibawah dua kotak teks tersebut akan ada tombol untuk diklik bila teks abstrak artikel ilmiah yang dimasukan telah siap untuk diuji lalu pada bawah tombol tersebut akan menampilkan angka berapa besar persen kemiripan yang terjadi antara dua teks abstrak artikel ilmiah tersebut.

A. Flask

Flask adalah *micro web framework* yang berbasis Python. Flask dirilis pada tahun 2010 lalu oleh Armin Ronacher. Meskipun usianya yang masih sangat muda, pengguna Flask sudah cukup sangat menyebar di kalangan para pengembang website berbasis python untuk para pemula[6].

Flask ini memiliki kesamaan dengan Django, meskipun Flask ini dikenal sebagai web framework untuk pemula, Flask juga bisa Anda gunakan untuk membuat website yang memiliki tingkat kerumitan yang lebih tinggi. Artinya, untuk membuat aplikasi profesional pun menggunakan Flask juga tetap bisa.

Sama halnya dengan metode *coding* dengan menggunakan web framework lainnya. Kita harus membuat struktur file yang rapi. Untuk membuat struktur tersebut, kita bisa membuat folder dan file dalam *flaskapp/*. Dalam *flashkapp/* ini, Anda buat folder *app/* yang menjadi tempat penyimpanan semua file. Kemudian, dari folder *app/* ini, Anda membuat kembali folder yang dinamakan *static/*. Pada folder inilah Anda mulai menyimpan file dalam folder yang dapat Anda tandai sesuai kebutuhan seperti css, javascript maupun gambar[6]. Sedangkan untuk menyimpan templates, Anda membuat folder baru di dalam folder *app/*. Yang bisa Anda beri nama *templates*. Kemudian, di dalam folder *app/* pula, Anda membuat file kosong yang diberi nama *routes.py*. File ini nantinya akan menjadi tempat program yang menjadi logika aplikasi seperti routing alamat URL. Berikut adalah contoh sederhana untuk struktur folder dan file pada flask[6].



Gambar 2.1 contoh struktur file pada flask[6]

B. Natural Language Toolkit

Natural Language Toolkit adalah *library* yang terdapat pada bahasa *python* yang sangat membantu untuk mengolah data teks dalam *Natural language Processing* (NLP). NLTK dapat melakukan tugas-tugas seperti *parsing sintaksis*, klasifikasi teks, dan *part of speech.* Implementasi dari pengerjaan tugas-tugas dapat dikombinasikan untuk pengerjaan tugas yang lebih kompleks.

Tabel 1 - Tabel 2.2 Modul NLTK

Task	Modul NLTK	Fungsi
Akses Corpora	nltk.corpus	Interface standar untuk corpora dan lexicon
Pengolahan string	nltk.tokenize, nltk.stem	Tokenizers, tokenizers kalimat, stemmers
penandaan bagian pada speech	nltk.tag	n-gram, backoff, Brill, HMM, TnT
Klasifikasi	nltk.classify, nltk.cluster	Decision tree, maximum entropy, naive Bayes, EM, k-means
Chunking	nltk.chunk	Regular expression, n-gram, named entity
Parsing / penguraian	nltk.parse	Chart, feature-based, unification, kemungkinan,
Evaluation metrics	nltk.metrics	Precision, recall, agreement coefficients

NLTK memiliki 4 tujuan utama[7]:

1. *Simplicity*, memberikan ilmu NLP dengan praktis secara mudah kepada pengguna, meyediakan kerangka kerja intuitif.
2. Konsisten, untuk memberikan kerangka kerja dengan *interface* yang konsisten.
3. *Extensibility*, struktur yang dapat menerima *software* baru dengan baik
4. *Modularity*, menyediakan komponen pada *tool* yang terdapat pada *library* dan dapat digunakan secara terpisah tnapa harus mengerti dengan *toolkit* lainnya.

C. *Natural language Processing*

Natural Language Processing (Pemrograman Bahasa Alami) adalah pembuatan program yang memiliki kemampuan untuk memahami bahasa manusia. Pada prinsipnya bahasa alami adalah suatu bentuk representasi dari suatu pesan yang ingin dikomunikasikan antar sesama manusia.

Natural language processing adalah upaya untuk mengekstrak lebih jauh representasi dari suatu teks bebas. Hal ini dapat dimasukkan secara kasar seperti mencari siapa melakukan apa kepada siapa kapan, di mana, bagaimana dan mengapa. NLP terkadang membuat penggunaan konsep-konsep linguistic seperti kata benda, kata kerja, kata sifat, dan lainnya dan struktur gramatikal (baik direpresentasikan sebagai ungkapan-ungkapan seperti frase nomina atau frase preposisional, atau hubungan ketergantungan seperti subjek dari- atau objek -dari).

Tujuan dari pemrograman bahasa alami adalah melakukan proses pembuatan model komputasi dari bahasa sehingga dapat terjadi suatu interaksi antara manusia dengan komputer dengan perantara bahasa alami[8].

D. Pra Proses (*Preprocessing*)

Menurut Zaman dan Winarko (2011) yang merujuk kepada Neto *et al* (2003) pada tahap pra proses (*preprocessing*) dilakukan penyiapan dokumen mentah menjadi dokumen atau representatif dokumenyang siap diproses untuk langkah selanjutnya. Pada tahp ini proses yang dilakukan antara lain membagi dokumen menjadi kalimat, *case folding*, menghapus *stopword*, melakukan proses *stemming* dan membagi dokumen menjadi kata (*tokenizing*)[8].

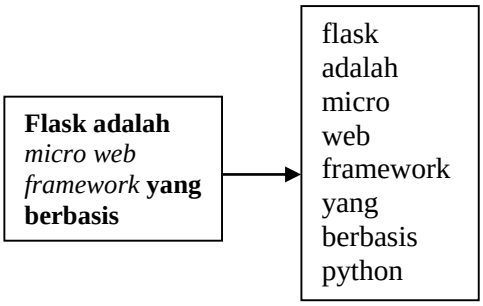
1. Pemecahan Kalimat

   Memecah dokumen manjadi kalimat-kalimat merupakan langkah awal tahapan *preprocessing*.

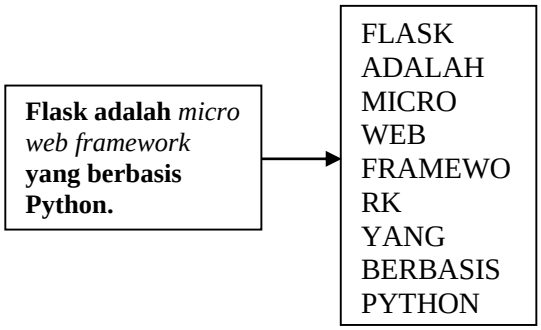
Pemecahan kalimat yaitu proses memecah string teks dokumen yang panjang menjadi kumpulan kalimat-kalimat. Dalam memecah dokumen menjadi kalimat-kalimat menggunakan fungsi split (), dengan tanda titik "." Tanda tanya "?" dan tanda seru "!" sebagai pemisah (*delimiter*) untuk memotong string dokumen.

2. *Case Folding*

Case folding adalah tahapan proses mengubah semua huruf dalam teks dokumen menjadi huruf yang alfabet yang ditentukan seperti, apabila peneliti menginginkan huruf dalam teks menjadi alfabet kecil maka menjadi kecil semua sebaliknya bila peneliti menginginkan huruf yang ada pada teks tersebut berukuran alfabet besar maka menjadi besar.



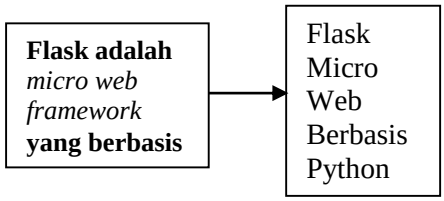
Gambar 2.2 Simulasi lowercase



Gambar 2.3 Simulasi uppercase

3. Penghapusan *Stopword*

Penghapusan *stopword* merupakan proses penghilangan kata *stopword*. *Stopword* adalah kata-kata yang sering kali muncul dalam dokumen namun arti dari kata-kata tersebut tidak deskriptif dan tidak memiliki keterkaitan dengan tema tertentu. Misalnya "di", "oleh", "pada", "sebuah", "karena" dan lain sebagainya.



Gambar 2.4 simulasi penghapusan stopword

4. *Stemming*

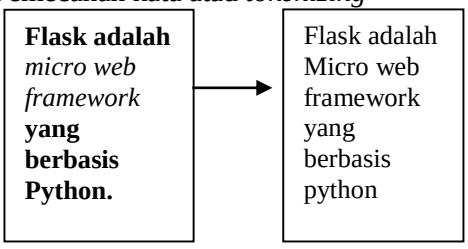
Menurut Zaman dan Winarko (2011) yang merujuk kepada Kurniawan (2003) *stemming* adalah proses pemetaan dari penguraian berbagai bentuk kata baik itu *prefix*, *sufix*, maupun gabungan antara *prefix* dan *sufix* (*confix*), menjadi bentuk kata dasarnya. *Stemming* pada penelitian ini menggunakan Algoritma Nazrief dan Adriani. Menurut Andita (2010) menyatakan bahwa algoritma stemming Nazarief dan Adriani 91996) dikembangkan berdasarkan aturan morfologi Bahasa Indonesia yang mengelompokkan imbuhan menjadi awalan (*prefix*), sisipan (*infix*), akhiran (*suffix*) dan gabungan awalan-akhiran (*confixes*). Algoritma ini menggunakan kamus kata dasar dan mendukung recording, yakni penyusunan kembali kata-kata yang mengalami proses *stemming* berlebih (Andita, 2010).

Algoritma *stemmer* yang diperkenalkan Nazrief dan Adriani didefinisikan sebagai berikut (Andita, 2010):

1. Di awal proses *stemming* dan setiap langkah yang selanjutnya dilakukan, lakukan pengecekan hasil proses *stemming* kata yang di input-kan pada langkah tersebut ke kamus kata dasar. Jika kata ditemukan, berarti kata tersebut sudah berbentuk kata dasar dan proses stemming dihentikan. Jika tidak ditemukan, maka langkah selanjutnya dilakukan.
2. Hilangkan *Inflenction Suffixes* (“-lah”’ “-kah”, “-ku”, “-mu”, atau “-nya”). Jika berupa *particles* (“-lah”, “-kah”, “-tah”, atau “-pun”) maka langkah ini diulang lagi untuk menghapus *Possesive Pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.
3. Hapus *Derivation Suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a
   a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihaous. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
   b. Akhiran yang dihapus (“-i”, “-an”, atau “-kan”) dikembalikan, lanjut ke langkah 4.
4. Hilangkan *derivation prefixes*
   a. Langkah 4 berhenti jika :
   i. Terjadi kombinasi awalan dan akhiran yang terlarang.
   ii. Awalan yang dideteksi saat ini sama dengan awalan yang dihilangkan sebelumnya.
   iii. Tiga awalan telah dihilangkan.
   b. Identifikasi tipe awalan dan hilangkan. Awalan terdiri dari dua tipe :
      i. Standar (“-di”, “ke-”, “se-”) yang dapat langsung dihilangkan dari kata.
      ii. Kompleks (“me-”, “be-”, ”pe”, “te-”) adalah tipe-tipe awalan yang dapat bermorfologi sesuai kata dasar yang mengikutinya.
   c. Cari kata yang telah dihilangkan awalannya ini didalam kamus kata dasar. Apabila tidak ditemukan, maka langkah 4 diulangi kembali. Apabila ditemukan, maka keseluruhan proses dihentikan.
5. Apabila setelah langkah 4 kata dasar masih belum ditemukan, maka proses *recording* dilakukan dengan mangacu pada aturan pemenggalan awalan stemmer Nazrief dan Adriani. *Recording* dilakukan dengan menambahkan karakter *recording* di awal kata yang dipenggal. Karakter recording adalah huruf kecil setelah tanda hubung (“-”) dan terkadang berada sebelum tanda kurug.
6. Jika semua langkah gagal, maka input kata yang diuji pada algoritma ini dianggap sebagai kata dasar.

5. Pemecahan kata (*Tokenizng*)

Pemecahan kata atau *tokenizing*



Gambar 2.5 Simulasi tokenizing

Tokenisasi adalah salah satu prose yang terjadi dalam *preprocessing text*, teks yang ingin di proses dalam algoritma pemograman *natural language processing* maka akan melalui setiap praproses tersebut, untuk mempercepat dan memudahkan teks yang ada untuk di proses dalam program.

Tokenisasi memisahkan kata-kata yang terdapat dalam sebuah teks menjadi kata-kata satuan, tokenisasi menjadikan karakter *whitespace* {menjadi sebagai pemisah kata-kata yang terdapat di dalam teks tersebut[9].

E. Cosine Similiarity

Cosine Similarity adalah ukuran kesamaan antara dua vektor n dimensi dengan mencari cosinus dari sudut antara dua vektor. Pada metode cosine similarity tidak melihat dari panjang pendeknya dokumen melainkan dari nilai term masing- masing. Berikut adalah persamaan dari metode Cosine Similarity [11]:

$$similarity = cos\,(A,B) = \frac{|A \cap B|}{\sqrt{|A|}\sqrt{|B|}}$$

Gambar 2.6 Rumus perhitungan cosine similiarity

A dan B adalah kata yang dihitung kemiripannya, yang telah di lakukan pemotongan karakter (ngrams) dari kata A dan B terlebih dahulu. Nilai kesamaan kosinus antara 0 dan 1. Dua vektor dikatakan sama jika membentuk sudut 00 atau nilai kosinusnya 1.

Langkah-langkah perhitungan similarity dengan metode cosine adalah:

1. Hitung jumlah irisan dari ngams kata A dan B:
   Intersection = $|A \cap B|$
2. Kemudian hitung semua produk menggunakan persaman:
   Product = $|A|.|B|$
3. Kemudian hitung tingkat kemiripan pada tiap kata dengan menggunakan persamaan yang terdapat pada gambar 2.6[10].

F. Whitebox and Blackbox Testing

Whitebox testing adalah metode pengujian berdasarkan barisan-barisan kode yang dibuat dalam prses pemograman, pada proses pengujian diharuskan mengetahui bahasa pemograman dan logika-logika yag ada didalam pemograman tersebut. Whitebox memiliki beberapa tahapan untuk menguji, yaitu [11]:

1. pengujian sesuai dengan logika bahasa pemograman.
2. pengujian melihat kode-kode sesuai dengan fungsi atau tidak.
3. Pengujian pada struktur data yang terdapat pada barisan kode

Sedangkan *Blackbox Testing* adalah metode pengujian berdasarkan fungsi yang ada pada sistem tanpa mengetahui barisan-barisan kode. Tujuan dari metode *blackbox* adalah menguji sistem untuk siap digunakan pada *user* sehingga tidak ada kesalahan yang dapat mengganggu jalannya sistem dan memastikan *user* menggunakan sistem yang dapat menangani kesalahan *user* dalam *input* yang dilakukan oleh *user*. Metode *Blackbox Testing* memiliki tujuan untuk menemukan beberapa hal, yaitu :

1. *Interface* yang tidak sesuai
2. Fungsi-fungsi pada *interface* sesuai kebutuhan
3. Kesesuaian performa pada sistem

Metode-metode diatas adalah metode pengujian yang secara tidak sadar sering digunakan oleh *developer* namun sangat penting untuk meningkatkan dan mengurangi kesalahan yang terjadi pada pembangunan sebuah aplikasi atau sistem.

III. HASIL DAN PEMBAHASAN

A. Hasil

Berikut adalah hasil dari pengujian pada aplikasi:

Tabel 2 - Tabel 4.1 Hasil Pengujian

Nama Proses	Proses	Kode	Hasil
Preprocessing	Menggantikan huruf yang berbeda-beda pada abstrak menjadi huruf kecil	Melakukan penyetaraan teks menjadi lower case, penghapusan tanda baca, pemisahan kata per kata,, menguraikan kata menjadi bentuk standar.	Mendapatkan abstrak artikel ilmiah yang telah menjadi huruf kecil dan tanda baca yang telah dihilangkan
Perancangan algoritma cosine similiairity	Membandingkan abstrak artikel ilmiah uji dengan abstrak artikel ilmiah acuan	Membagi teks yang telah dilakukan preprocessin, perhitungan cosine similiarity, melakukan perbandingan pada teks abstrak.	Mendapatkan hasil perbandingan antara teks uji dengan teks acuan dan dijadikan dalam bentuk persentase
Pengujian cosine similiarity	Menguji model dengan data teks dalam user interface.	Memuat model, memasukan teks, membandingkan teks, memberikan hasil persentase.	Mengetahui seberapa besar persentase yang didapatkan setelah membandingkan kedua teks

Pengujian telah dilakukan pada penelitian ini dan memiliki tiga proses. Proses pengujian meliputi *preprocessing, training model, testing app.* Proses pengujian dapat dicapai dengan metode *whitebox* metode yang digunakan untuk menguji setiap bari-baris kode yang telah dituliskan sehingga diharapkan sesuai dengan proses tersebut.

Pada tahap implementasi akan dijelaskan barisan kode-kode dan perhitungan *cosine similiarity* serta menampilkan hasil yang dikeluarkan pada setiap bari kode-kode yang ditulis. Penjelasan barisan kode-kode dilakukan dengan membahas setiap barisan kode-kode sesuai fungsi-fungsi yang digunakan didalam proses pembuatan aplikasi.

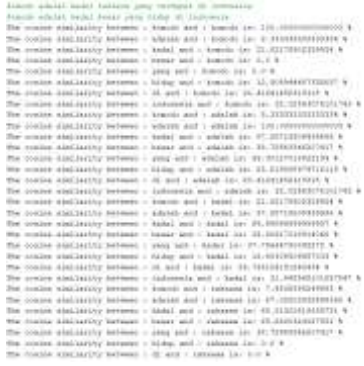
B. Analisis

Hasil analisa yang dilakukan oleh peneliti adalah menggunakan perhitungan *cosine similiarity* sebagai perhitungan yang digunakan untuk menghitunga kemiripan pada sebuah teks abstrak. Peneliti menggunakan *cosine similiarity* sebagai perhitungan untuk deteksi kemiripan dikarenakan perhitungan *cosine similiarity* cocok untuk mendteksi kemiripan dikarenakan kalimat-kalimat yang terdapat pada sebuah teks abstrak dapat memiliki kemiripan meski berbeda secara penulisan ataupun tata bahsaa oleh karena itu perhitungan *cosine similiarity* cocok digunakan dan dibantu dengan *library python* yang dapat membantu dalam mengurangi kesalahan yang terjadi.

Kesalahan yang dapat terjadi adalah kalimat-kalimat pada teks abstrak artikel ilmiah tidak menggunakan kata baku atau menggunakan imbuhan yang dapat mengurangi keakuratan dalam mendeteksi kemiripan sehingga dibantu dengan *library python* yang bernama *Sastrawi* yang berfungsi untk menjadikan kata-kata yang terdapat pada kalimat teks itu tidak baku menjadi kata baku yang tidak memiliki imbuhan.

C. Perancangan

Perancangan yang telah dilakukan oleh peneliti adalah menggunakan perhitungan *cosine similiairity* yang dapat digunakan sebagai program untuk mendeteksi kemiripan sebuah teks. Teks abstrak artikel ilmiah yang dapat dideteksi kemiripan dengan teks abstrak lainnya sebagai acuan.



Gambar 4.1 kode cosine similarity untuk kata

Kode yang digunakan untuk mendeteksi kemiripan karakter-karakter pada teks tersebut dengan menggunakan deteksi pada setiap huruf yang terdapat pada teks tersebut.

Gambar 4.2 hasil dari mendeteksi

Hasil yang dapat dilihat pada gmabar 4.2 adalah hasil dari mendeteksi kemiripan pada kata-kata yang terdapat sebuah teks uji coba lalu menghitung *cosine similiairty* dari kedua kata tersebut lalu dikali dengan serratus persen dan hasil yang dikeluarkan seperti pada gambar 4.2 dengan setiap kata ditampilkan dan dengan kata apa itu

Gambar 4.3 Model Rancangan website

Hasil rancangan *website* ini adalah hasil yang sangat sederhana untuk menampilkan *user interface* untuk menggunakan aplikasi deteksi kemiripan sebuah teks abstrak artikel ilmiah ini dan diharapkan mudah dalam menggunakan sehingga tidak membuat kesulitan dalam penggunaannya.

D. Implementasi

Gambar 4.4 Kode integrasi progam dengan website

Perancangan *website m*enggunakan *library python flask* yang dapat membantu dalam pembangunan *website* dan mengintegrasikan dengan program *python* yang dapat berbeda *file*. Dan pada kode pada gambar 4.4 terdapat kode yang melakukan *preprocessing* yang dibantu dengan *library python regex.*

Gambar 4.5 Kode Perhitungan cosine similiarity

Kode yang terdapat pada gambar 4.5 itu memiliki berfungsi yang membantu perhitungan *cosine similiarity* yaitu fungsi menghitung berapa banyak kata yang terdapat pada kalimat dalam teks abstrak artikel ilmiah yang dimasukkan pada *form website* dan digunakan untuk perhitungan yang mendeteksi kemiripan.

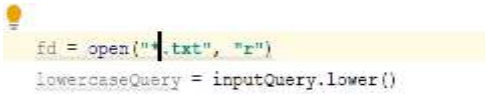
Gambar 4.6 user interface deteksi kemiripan

Tampilan hasil dari baris kode pada gambar 4.3 hasilnya dapat dilihat pada gambar 4.6 *form* yang terdapat *user interface* akan digunakan untuk memasukan teks abstrak artikel ilmiah yang diuji pada form teks satu dan form teks dua digunakan untuk memasukan teks abstrak artikel ilmiah yang menjadi acuan untuk mengetahui berapa persentase kemiripan yang terjadi diantar kedua teks tersebut.

Gambar 4.7 Hasil dari deteksi kemiripan

Teks satu dan teks dua memiliki kemiripan sebesar 75.00% form yang telah disi akan dikirimkan ke program *python* lalu akan diproses menggunakan perhitungan *cosine similiarity* dan hasil dari perhitungan akan dikirim dan ditampilkan seperti gambar 4.7.

E. Evaluasi



Gambar 4.8 kode masukan teks

Barisan kode tersebut dapat diubah pada bagian *input* pada kode yang terdapat pada gambar 4.4 sehingga data yang digunakan sebagai acaun adalah teks abstrak artikel ilmiah yang sudah ada dalam *database* dan tidak perlu menggunakan teks acuan lagi.

IV. SIMPULAN

Kesimpulan dari hasil pembuatan aplikasi deteksi kemiripan berbasis *websit* adalah sebagai berikut:

1. Pembangunan aplikasi ini menggunakan perhitungan *cosine similiarity* untuk deteksi kemiripan teks abstrak artikel ilmiah.
2. Kesalahan dalam memasukan teks ke dalam form dapat mempengaruhi hasil deteksi.
3. Kesalahan *Preprocessing* data teks dapat memengaruhi perhitungan *cosine similiarity*.
4. Aplikasi deteksi kemiripan abstrak artikel ilmiah menggunakan bahasa indonesia
5. Penggunaan *library* NLTK dan Sastrawi dapat mempengaruhi hasil deteksi kemiripan

DAFTAR RUJUKAN

[1] "3 Library Python Terbaik Untuk Data Science – Belajarpython – Situs Open Source Tutorial Pemrograman Python Bahasa Indonesia.", *Belajarpython.com*, 2020. [Online]. Available: https://belajarpython.com/2018/09/3-library-python-terbaik-untuk-data-science.html. [Accessed: 07- Aug- 2020].

[2] "Plagiarism dalam penulisan ilmiah". BennyLie , November 2017 , Online , https://docplayer.info/32028080-Plagiarism-dalam-penulisan-artikel-cAazaQzilmiah-kecurangan-di-dunia-ilmiah-1-6.html. [Accessed: 24- Mar- 2020].

[3] "Dugaan plagiarisme di UNJ: 'Pelaku ingin naik pangkat dan dipandang tinggi" , Admin , 6 September 2017 , Online , https://www.bbc.com/indonesia/indonesia-41161834 . [Accessed: 24- Mar- 2020].

[4] Deteksi Plagiat Dokumen Tugas Daring Laporan Praktikum Mata Kuliah Desain Web Menggunakan Metode Naive Bayes. (2016). *Nusantara Journal of Computers and its Applications*, 2(1), p.1.

[5] A. Yudhana, A. DjalilDjayali and S. Sunardi, "Sistem Deteksi Plagiarisme Dokumen Karya Ilmiah Dengan Algoritma Pencocokan Pola", *e-journal.unmul.ac.id*, 2017. [Online]. Available:

http://dx.doi.org/10.30872/jurti.v1i2.917. [Accessed: 24- Mar- 2020].

[6] L. Polepeddi, "Pengenalan Framework Flask pada Python", *Code Envato Tuts+*, 2020. [Online]. Available: https://code.tutsplus.com/id/tutorials/an-introduction-to-pythons-flask-framework--net-28822. [Accessed: 14- Aug- 2020].

[7] "Menggunakan NLTK untuk Pemrosesan Teks", *School of Computer Science*, 2020. [Online]. Available: https://socs.binus.ac.id/2018/08/09/menggunakan-nltk-untuk-pemrosesan-teks/. [Accessed: 17- Jun- 2020].

[8] "Text Preprocessing | INFORMATIKALOGI", *INFORMATIKALOGI*, 2020. [Online]. Available: https://informatikalogi.com/text-preprocessing/. [Accessed: 04- Jun- 2020].

[9] Belajarpython.com. 2020. *3 Library Python Terbaik Untuk Data Science – Belajarpython – Situs Open Source Tutorial Pemrograman Python Bahasa Indonesia.*. [online] Available at: <https://belajarpython.com/2018/09/3-library-python-terbaik-untuk-data-science.html> [Accessed 6 April 2020].

[10] M. Fachrurrozi, and A. A. Manik, "Perbaikan Ejaan Kata pada Dokumen Bahasa Indonesia dengan Metode Cosine Similarity Perbaikan Ejaan Kata pada Dokumen Bahasa Indonesia dengan Metode Cosine Similarity",Seminar Nasional Rekayasa Komputer danAplikasinya,Universitas Andalas, 2016.

[11] "3 Library Python Terbaik Untuk Data Science – Belajarpython – Situs Open Source Tutorial Pemrograman Python Bahasa Indonesia.", *Belajarpython.com*, 2020. [Online]. Available: https://belajarpython.com/2018/09/3-library-python-terbaik-untuk-data-science.html. [Accessed: 07- Aug- 2020].